Trajectory and Environment Modeling for Human Robot Collaboration Predictive Collision Detection

A Thesis

Presented to

The Faculty of the Department of Mechanical and Aerospace Engineering

Within the Herbert Wertheim College of Engineering

University of Florida

In fulfillment of the Honors Thesis Requirement for Graduation With Honors

by

Gabriel Streitmatter

Under the advisement of Faculty Member

Dr. Gloria Wiens

April 17, 2020

Abstract

Robots and humans closely working together within dynamic environments must be able to continuously look ahead and identify potential collisions within their ever-changing environment. To enable the robot to act upon such situational awareness, its controller requires an iterative collision detection capability that will allow for computationally efficient Proactive Adaptive Collaboration Intelligence (PACI) to ensure safe interactions.

In this thesis, an algorithm is developed to evaluate a robot's trajectory, evaluate the dynamic environment that the robot operates in, and predict collisions between the robot and dynamic obstacles in its environment. This algorithm takes as input the joint motion data of predefined robot execution plans and simulated sensor data on the predicted joint motions of the human and obstacles. It constructs sweeps of the robot's and human's instantaneous poses, and obstacles throughout time. The sweeps model the trajectories as point clouds containing all locations occupied by either body and the time at which they will be occupied. To reduce the computational burden, Coons patches are leveraged to approximate intermediary positions between the instantaneous poses. Overlaying temporal mapping of the sweeps reveals anticipated collisions that will occur if the robot or the human do not proactively modify their motion. The algorithm is designed to feed into a segmentation and switching logic framework and provide real-time proactive-n-reactive behavior for different levels of human-robot interactions, while maintaining safety and production efficiency. Run time data collected on this algorithm for a significant number of randomly generated test cases at various degrees of modeling precision suggest that the algorithm can accurately predict future collisions in under 30 ms.

Keywords: Predictive Collision Detection, Swept Volume Interference, Coons Patches

Introduction

Automation of manufacturing processes over the past few decades has yielded enormous benefits in efficiency and quality. This was accomplished by increasing speed and precision by unlocking new capabilities through introduction of innovative material handling devices and equipment along with computer enabled interconnectivity of heterogeneous sensor suites and controls. As demands on manufacturing continue to evolve, however, more is required. Products are experiencing quicker life cycles and increased customization. This means that manufacturing processes must, in addition to maintaining efficiency and quality, be more flexible and adaptable to meet rapidly changing demands. This can be accomplished by developing human-robot collaboration (HRC) methodologies; taking advantage of the speed, power, and precision of robots as well as the creativity and adaptability of humans [1]. Teams of humans and robots are uniquely capable of meeting stringent and rapidly changing requirements by engaging in HRC and combining their talents.

To do this, robots must be flexible enough to operate safely around humans actively working within fenceless environments. This further elevates challenges as safety concerns surround unrestricted operation of robots near humans. Before HRC can be implemented, robots must learn to predict and avoid collisions with humans in real time.

In previous work, robot trajectories were subdivided into a series of segments. The segments were highly customizable to specific demands on the robot during operation. With this approach, the robot was able to respond to obstacles according to localized features of the segment of the trajectory under execution rather than the generalized trajectory as a whole [2]. Next, an algorithm for predictive collision detection was developed to enable a robot to forecast its motion amongst moving ellipse shaped object profiles to identify collisions. Coons patches were leveraged to approximate the robot's position throughout time as a swept volume [3]. Finally, the algorithm was extended to model the articulated, multi-link body trajectories of the human to predict collisions between a robot and human. In this work, the algorithm was further optimized for increased computational speed and modeling accuracy [4]. This thesis seeks to combine the latter two approaches and outline them in greater detail. The overall goal of this approach is to provide a robot cognizance of its surrounding dynamic environment and an ability to understand predictions of how the environment will evolve in the immediate future. Additionally, any methods used must be applied quickly and iteratively as the robot’s environment changes. Thus, in addition to accurate modeling, a key goal for this work is to develop an algorithm that minimizes computational time and can be implemented in real time. This, in effect, gives the robot a situational awareness that is essential to intelligent dynamic response.

The work presented in this thesis contributes to a multi-university/industry/national laboratory collaborative research effort to develop HRC techniques for sensing of dynamic environments, forecasting change in the environment, and enabling robots to respond safely and productively [5]. The algorithm presented in this thesis will contribute to an HRC Proactive Adaptive Collaborative Intelligence (PACI) module used to control robot motion and operation [2].

Related Work

One of the simplest and fastest ways of performing a predictive collision detection is to employ “*Multiple Interference Detection”.* This is done by selecting a representative set of configurations (poses) throughout the robot’s trajectory at which to check for interference. While this method is generally much faster than most other methods, it runs the risk of missing a collision due to an insufficient number of interference checks [6]. There are various techniques to help determine how often a trajectory should be checked for collision. One technique is to use the velocities of the obstacles and the distance between them to predict the earliest possible time at which a collision could happen. This prediction can be used to determine when to check for interference next [7].

One of the most common methods is to investigate “*Swept Volume Interference*”. In this method, a volume is generated by sweeping an object along a trajectory and including in the volume all points occupied by the object at any given time. This can be done for multiple objects. If the volumes generated by any objects intersect and the times at which the objects pass through the intersection points in the volumes match, then a collision has been identified [6]. While this solution is attractive in that it doesn’t suffer from the higher probability of missing collisions due to insufficient sampling that plagues “*Multiple Interference Detection*”, it is computationally expensive and thus difficult to implement in real time or even near real time. For complex objects or sweeps that employ small step sizes, floating point error can prove to be inhibitive for even dedicated processors, let alone processors that are also controlling a robot [8]. For the purposes of predictive collision detection in dynamic environments in which the validity of a robot trajectory must be re-evaluated every time the environment changes, faster algorithms are required.

Attempts have been made to improve the computational efficiency of *Swept Volume Interference*. One proposed approach to reducing the computational cost of sweeping the volume is to work strictly with objects modeled as complex polyhedra. Since collision detection is much more efficient with convex polyhedra, algorithms have been developed to decompose concave meshes into a number of convex volumes. The convex volumes can then be used to generate a swept volume to use in a collision detection algorithm with a lower computational cost than if the concave volumes were used [9]. This approach is utilized in [10] to represent a robot's swept volume during a trajectory. This approach allows collisions to be detected through swept volumes and is able to reduce computational time.

Another approach compares sets of points that are contained within the respective swept volumes of multiple objects to identify collisions. A common method of constructing this set is to employ Point Membership Classification (PMC) which identifies weather a point falls within an object's boundary, on its boundary, or exterior to its boundary [11]. The points selected to represent each volume are minimized by only evaluating the boundary of the object and by decreasing the density of the points. This approach capitalizes on the tradeoff between computational efficiency, which can be maximized by minimizing the point density, and quality of the collision detection, which increases with the point density [12].

One hybrid approach, in which a few of the aforementioned techniques are synthesized is employed in [13], where a pre-existing 3D surface is sampled as it moves along its trajectory at a high frequency such that an approximate swept volume is created. While this approach is faster than computing a swept volume, it requires the pre-existence of a 3D surface, and a large number of samples to construct the surface.

The algorithm presented in this thesis addresses the need for a higher speed collision detection algorithm that can operate on much more basic predictions of the human's location and orientation throughout time. By implementing Coons patches to interpolate the human's position throughout time, fewer samples are required to construct a representative swept surface, greatly improving the computational efficiency.

Methods

The developed algorithm assumes as its only input the preplanned trajectory of the robot, in terms of the robot's joint angles, and predicted motion derived from simulated sensor data describing joint angles of a human's articulated multi-link body at various instants in time. This simulated data is in the format expected and to be provided by the project's research collaborators developing a sensor suite and prediction methodology. From this input, forward kinematics techniques are used to define boundary points outlining the robot and the human at each predicted instant in time. Without loss of generality, a serial manipulator with all revolute joints will be assumed whenever reference to a robot is made. Since the linkage geometries are assumed to be constant, the translation matrices are constant. If cylindric or prismatic joints were present in the robot, the translation matrices would not be constant. This would be a simple extension, however, but it is not necessary in that the robot available for testing this algorithm could not make use of such an extension.

These boundary points, once defined by the forward kinematics, are then used to create boundary surfaces that envelop the swept volumes of the robot and the human. The proposed method seeks to combine the computational efficiency of "*Multiple Interference Detection"* with the guarantee of identifying all collisions offered by "*Swept Volume Interference*" methods. In other words, the algorithm provides continuous collision detection at a low computational cost by interpolating between a set of lower frequency samplings to create a continuous surface. Furthermore, since the swept volume is described only by the boundary surface, points within the surface do not need to be characterized, greatly reducing computer memory and computational costs. Simulations of the robot and human in a workspace are used to evaluate effectiveness and computational efficiency of the algorithm.

Overview of the Approach

To demonstrate the approach, a human and a robot will be modeled in a workspace. The simulated testing utilized in this research and in the overall project at large is completed with a multi-axis tabletop robot arm that in its present form does not have collaborative robot (cobot) capabilities. Such robots are largely used for small, tabletop assembly and inspection applications, the human model selected to interface with such a robot is a seated worker. Thus, only the waist up of the human was modeled. The formulation of both the human and the robot will be developed in this thesis in parallel. Let the workspace be defined as a 3D (XYZ) discrete cartesian space. The distance between each adjacent point will be called the grid size. A preliminary grid size must be set to determine coarseness of the grid on which the human and robot swept volumes are to be compared. The method for defining each swept volume does not initially conform to this grid and does not conform to the spacings used for other swept volumes. The purpose of the grid is to provide a means for direct comparison between multiple swept volumes. After each swept volume is created, all points defining the swept volume are relocated to the nearest grid location. A fine grid size will result in an increase in computational time but will a yield more accurate representation of the swept volume. Once the grid size is selected, the remainder of the algorithm executes three general steps. These steps will be briefly explained in this section and outlined in detail in subsequent sections.

The first step is the generate boundary curves around each object to be modeled. These curves represent the location of each object at a few poses throughout time. For modeling the robot, the poses of the robot throughout time are determined directly from the input joint angles through forward kinematics and are positioned such that they create a boundary around the robot (*Figure 1 Left*). As the robot moves to new orientations, the boundaries of all orientations will be patched together to create a surface. To account for motion of the robot in a direction parallel to the previously defined boundary, a second boundary around the robot is also created in the orthogonal direction (*Figure 1 Right*).

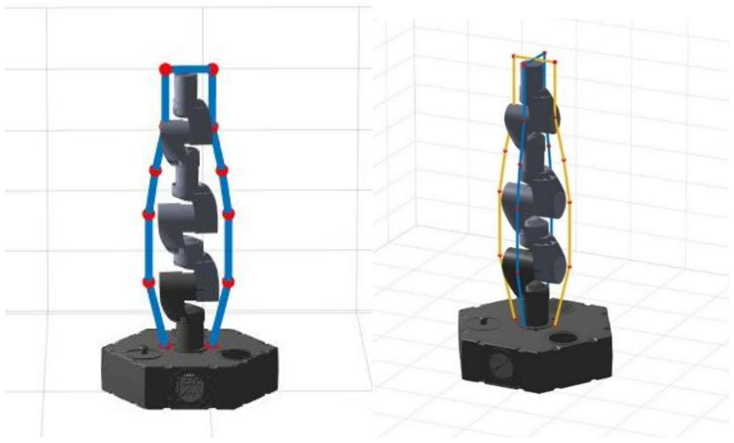


Figure 1. Boundary curves for modeling the robot in two orthogonal directions

Boundary curves for the human are defined next. For the purposes of this work, it is assumed that the initial orientation of the human is with their arms down, facing the robot. The algorithm however can start at arbitrary joint angles on the human. This initial orientation must simply be expressed in relation to the initial orientation assumed in this work. As more information is obtained from project's research collaborators about their human-obstacle motion predictor's data structure to serve as input data to the algorithm of this thesis, an initial preliminary step of the algorithm can be to evaluate the input to define these joint angles. As with the robot, multiple orthogonal boundary curves are required to ensure that lateral motion is contained within the swept volume. The first curve is defined in a plane facing the direction normal to the initial orientation of the human (*Figure 2 Left*). To account for motion in the orthogonal direction, since the torso, neck, and arms of the human do not fall in line, three different curves must be defined. One curve outlining the side profile of the human's torso, head and neck, and the other two curves outlining the side profile of each arm (*Figure 2 Right*). This defines a 3-D boundary of the human at one instance in time.

All curves used to define the human will result in their own swept surface. By overlaying all generated surfaces, the ones that represent the further extent of the modeled volume will define the outer surface of the swept volume, which will be used for the final collision check. For instance, if the arm were to rotate such that the direction of motion of the arm was entirely in the normal plane of the initial position of the human, the orthogonal boundary curve, when overlaid with other boundary curves would define the outermost surface. If the arm were to rotate such that the direction of motion of the arm was entirely in the orthogonal plane of the initial position of the human, the normal boundary curve would define the outer most surface. For situations between these two extremes, both boundary curves are required to give full representation of the locations occupied by the arm as it travels.

As with the robot, this process is repeated for multiple instances in time throughout the human's trajectory. To account for uncertainty in future predictions and safety considerations, safety factors are built into the dimensions of the boundary curves such that the boundary curves grow with time. The growth can be defined as a function of the body part of the human. For example, if a collision with the head is of greater safety concern than the arm, the dimensions of the head can be set to expand more rapidly than those of the arm.

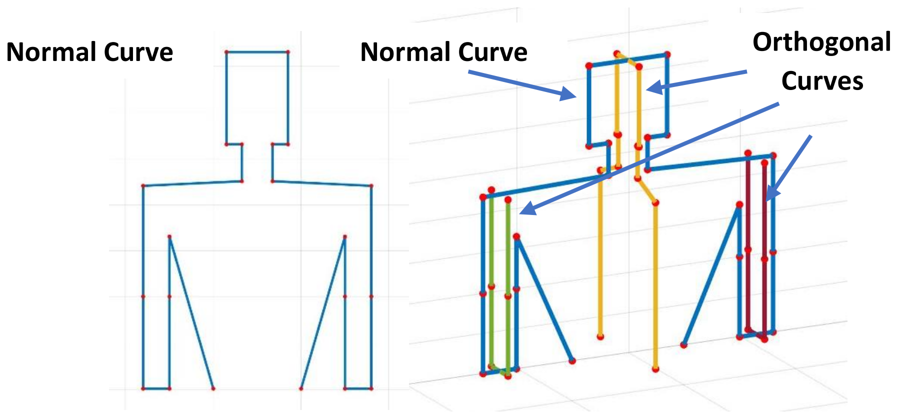


Figure 2. Boundary curves for modeling the human in two orthogonal directions

In the second step, discrete Coons patches [14] are used to define surfaces between each orientation throughout time. The patches provide an efficient expression for generating point clouds that fill the gaps between the given predicted positions of the human and robot. A patch is defined to model the motion of one straight segment of the robot or human as it moves between two poses. Patches are made to describe the cartesian positions of points occupied by the object in motion, and the time at which each position is occupied. These patches are key to maximizing computational efficiency by eliminating the need for completing forward kinematics at a high resolution and sampling frequency.

In the third step, initial and final conditions of the human and robot are implemented to generate surfaces at the beginning and end of the sweep. This step wraps a surface around the initial and final volumes of the human and robot. The resulting swept volumes can finally be compared with each other to identify spatiotemporal intersection of the volumes.

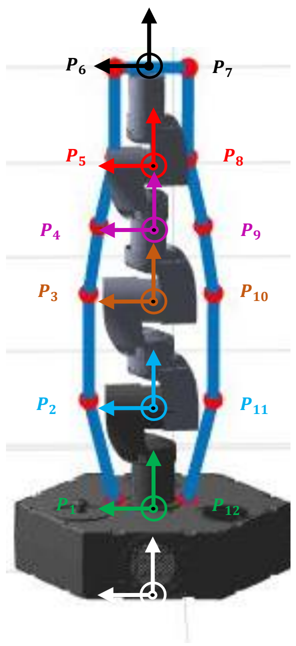
Step 1: Definition of the Boundary Curves with Forward Kinematics

The kinematics of the robot are described by a series of revolute joints. To describe the location of a point on the robot in a fixed coordinate system, located at the base of the robot, after arbitrary rotations of each revolute joint along the robot, intermediary coordinate systems are established with their origin locations fixed at the center of each joint. The location of these systems with respect to each other and with respect to an overall fixed system are described with forward kinematics. Specifically, transformation matrices are used to describe relative distances between two systems, and rotation matrices are used to rotate the systems about their corresponding joints. These matrices are defined in Denavit Hartenberg notation as done in [15].

To locate the position in the fixed coordinate system of any point on the robot after arbitrary rotations about each joint, a series of transformation and rotation matrices are used to translate points from the system of interest through each transformation and rotation of preceding systems to the fixed system. Consider a point, P, with a known position in the coordinate system *n*, ${}^{n}P$. To locate ${}^{n}P$ in the coordinate system of the previous joint, *n-1,* the point must be pre multiplied by a translation matrix, , ${}^{n-1}_{\ \ n}T$, that describes points in system *n* as seen from system *n-1.* A rotation about the joint in *n* will also affect the location of the point. To account for rotations, a rotation matrix rotating about joint *n*, R_n, must also be pre multiplied. A generalized formula for translating a point in the *n* system through an arbitrary number of translations and rotations is presented in (1) where ${}^{f}_{1}T$ is the translation from system one to the fixed system and ${}^{f}P$ is the location of point *P* as seen from the fixed system.

$$ {}^{f}P = {}^{f}_{1}T \ R_1 \left[\prod_{i=1}^{n-1} {}^{\ \ \ i}_{i+1}T \ R_{i+1} \right] {}^{n}P \qquad (1) $$

Using (1), boundary points can be defined at each joint to define a boundary curve around the robot. In Figure 3, the coordinate systems and points used to define one of the two boundary curves of the robot at one particular pose are shown. For all coordinate systems, the Z axis is in the upward direction, the X axis points to the left, and the Y axis points out of the page. The fixed coordinate system is shown in white. Each of the labeled points, P_1 through P_{12} are used to create the boundary. Figure 3 is color coded in that the color of the point label corresponds to the color of the triad used to define the coordinate system. For example, P_1 and P_{12} correspond to coordinate System 1. Each point's location is known in the coordinate system it corresponds to. The green, blue, brown, purple, red, and black triads represent coordinate systems one through six respectively. An example equation transporting P_7 from coordinate system six (${}^{6}P_7$) to the fixed coordinate system (${}^{f}P_7$) is given below. The transformation of each point used to define the boundary follows this pattern. The orthogonal boundary curve can be described in the same way, only with points defined along each coordinate system's Y axis rather than its X axis. Since all axis are revolute joints, the only variable in (2) are the rotation matrices, ***R,*** which vary as a function of the supplied joint angle.



Fixed System	⬜
System 1	🟩
System 2	🟦
System 3	🟫
System 4	🟪
System 5	🟥
System 6	⬛

Figure 3. Boundary points used to define a boundary curve for the robot

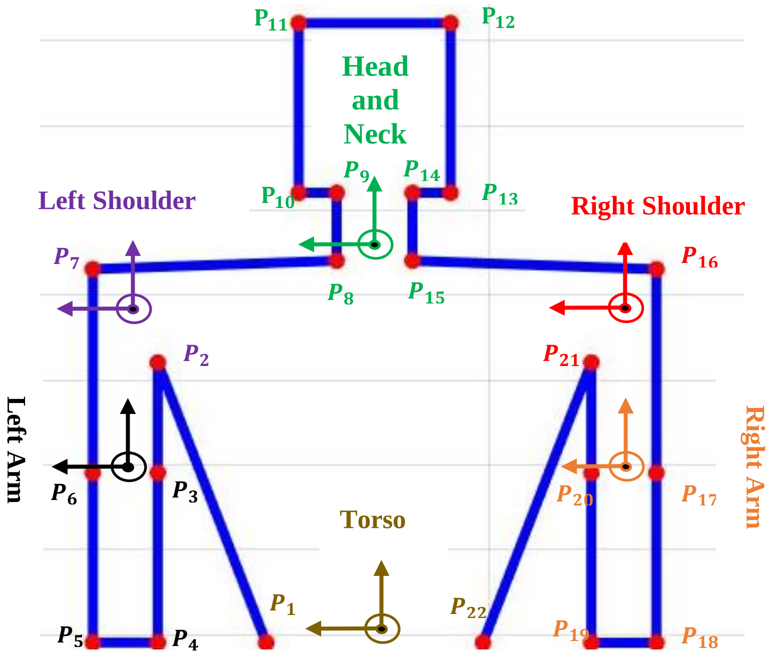
Thus, as the robot traverses its trajectory and the joint angles change, (2) remains valid and is able to describe the location of the robot at any set of joint angles. All equations and matrices locating of each point demarcated in figure 3 are shown in appendix A.

$$ {}^{f}P_7 = {}^{f}_{1}T\,R_1\,{}^{1}_{2}T\,R_2\,{}^{2}_{3}T\,R_3\,{}^{3}_{4}T\,R_4\,{}^{4}_{5}T\,R_5\,{}^{5}_{6}T\,R_6\,{}^{6}P_7 \qquad (2) $$

The kinematics of the human are defined by a torso joint with three axes of rotation, two shoulder joints with three axes of rotation each, two elbow joints with one axis of rotation each, and a neck joint with three axes of rotation. As done previously, (1) can be used to determine the location of all of the points required to create a boundary curve of the human. The coordinate systems used to define the human's kinematics are shown in Figure 4. The "Head and Neck" and "Torso" systems, shown in green and brown respectively, are both free to rotate about all three axes. The "Left Shoulder" and "Right Shoulder" systems, shown in purple and red respectively, are also both free to rotate about all three axes. The "Left Arm" and "Right Arm" systems, shown in black and orange respectively, are both free to rotate about their X axes. Points in each coordinate system are shown, color coded to indicate which system they belong to. These points define the boundary curve for the human and can be located in the fixed coordinate system with (1). For example, P_5 can be located with (3), and P_{13} can be located with (4). The boundary curves used to outline the front profile of the human will be referred to as the normal boundary curves. As done in appendix A, all equations and matrices used to locate of each point demarcated in figure 4 are shown in greater detail in appendix B.

$$ {}^{f}P_5 = {}^{f}_{1}T\ R_{1x}R_{1y}R_{1z}\ {}^{1}_{2}T\ R_{2x}R_{2y}R_{2z}\ {}^{2}_{3}T\ R_{3x}\,{}^{3}P_5 \tag{3} $$

$$ {}^{f}P_{13} = {}^{f}_{1}T\ R_{1x}R_{1y}R_{1z}\ {}^{1}_{6}T\ R_{6x}R_{6y}R_{6z}\,{}^{6}P_{13} \tag{4} $$



Fixed System	⬜
System 1	🟫
System 2	🟪
System 3	⬛
System 4	🟥
System 5	🟧
System 6	🟩

Figure 4. Boundary points used to define a boundary curve for the human

To define the side profiles of the human, a similar approach is utilized. For these, instead of drawing the boundary curve in the X-Z plane of each relative coordinate system, the boundary curve is drawn in the Y-Z plane of each system. The side profile of the human is constructed with three boundary curves. This set of boundary curves will be referred to as the orthogonal boundary curves. The side profile consists of a “Head Neck and Torso” section, and two “Arm Sections”. Each joint in these sections have the same rotation capability as the corresponding joint in the X-Z plane boundary curves. The same approach utilized before can be applied to find the location of each labeled point in the fixed coordinate system. See Appendix C for pertinent equations.

To build in a factor of safety and account for uncertainty in the predicted location of the human over time, the distance between these boundary points and the joint they correspond to is increased by the product of a scaling factor and the amount of time between the prediction and the execution time of the algorithm. The scaling factors for each body part are defined individually and can thus be set to increase at different rates. Definition and use of scaling factors is shown in appendix B.

For instance, the head can be set to expand at a higher rate in proportion to the time elapsed than the arms. From the side profile, the chest can be set to expand quickly as well. Figure 5 shows the human in the same pose but at different times. The blue line represents the initial size of the human while the yellow line represents the enhanced size of the human at some later time. There is a total of nine independently expanding entities: the head, the neck, the left shoulder, the right shoulder, the left upper arm, the right upper arm, the left forearm, the right forearm, the waist and the torso. Boundary points connected to two different expanding entities are set to use the larger of the two expansion factors to ensure the safest standard is used.

Points were set to expand in directions that would promote continuity of the boundary curve. For instance, points on the top of the head expanded upward and outward while points on the bottom of the head only expanded outward. Points on the top of the shoulders expanded in a radial direction away from the shoulder, and points on the bottom of the shoulder expanded laterally. Points on the arms expanded laterally outward, and points on the torso and waist expanded laterally outward. This same expansion procedure was also implemented for the side profile of the human's torso, neck, head, shoulders, arms and forearms. Each point on the side profile expanded in the same manner as similar points on the normal profile. For example, the side profile of the top of the human's head expanded upward and outward just as they did in the normal profile.

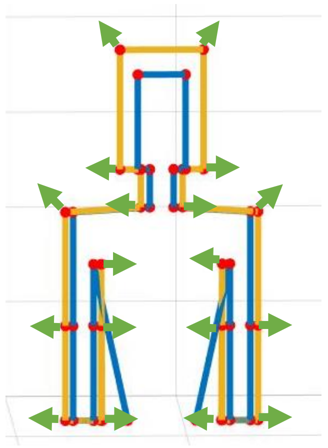


Figure 5. Expansion of boundary points used to account for increased uncertainty as the length of time of the forecasted prediction increases

These boundary curves are generated iteratively for each pose of the robot and human contained in the data. Up to this point, the boundary curves are simply cartesian points. In the next step, as the Coons patches are implemented, these points are connected to define points along the boundary curves. Figure 6 shows these boundary curves at various instances along the human's (*Right*) and robot's (*Left*) trajectories.

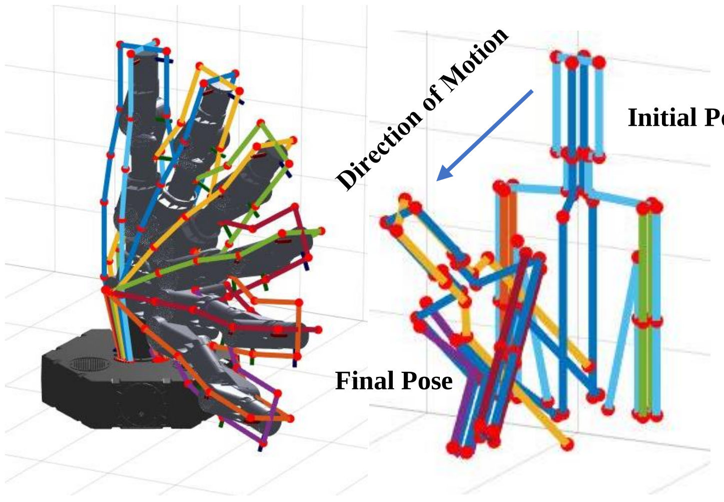


Figure 6. Boundary curves defined for the robot and the human thorughout multiple instances in their respective trajectories. For the sake of visual clarity, only the initial and final poses of the human were used in this figure. Intermediary poses however are used for the rest of this work.

Application of Coons Patches to Connect the Boundary Curves

With the boundary curves defined by specific points at the corners, the next step is to generate surfaces between them. This operation is completed with discrete Coons patches. An individual patch is created for each straight segment of the human as the segment travels between two different orientations. The boundary curves defined previously must now be put into a format that the discrete Coons patch formulation applies to. Example boundary curves for a Coons patch defined for one segment of the human head are shown in Figure 7. Four boundary curves are used to define each patch (*Figure 7 left*). The first boundary curve (*Curve 1*) follows a line between a point on the human at an instant in time and the same point at the next (second) instance in time. The second boundary curve (*Curve 2*) follows a line between the first point on the human at the second instance in time and a second point on the human at the second instance in time. The third boundary curve (*Curve 3*) follows a line between the second point at the second time and the same point on the human at the first time. The fourth boundary curve (*Curve 4*) follows a line between the second point at the first time and the first point at the first time.

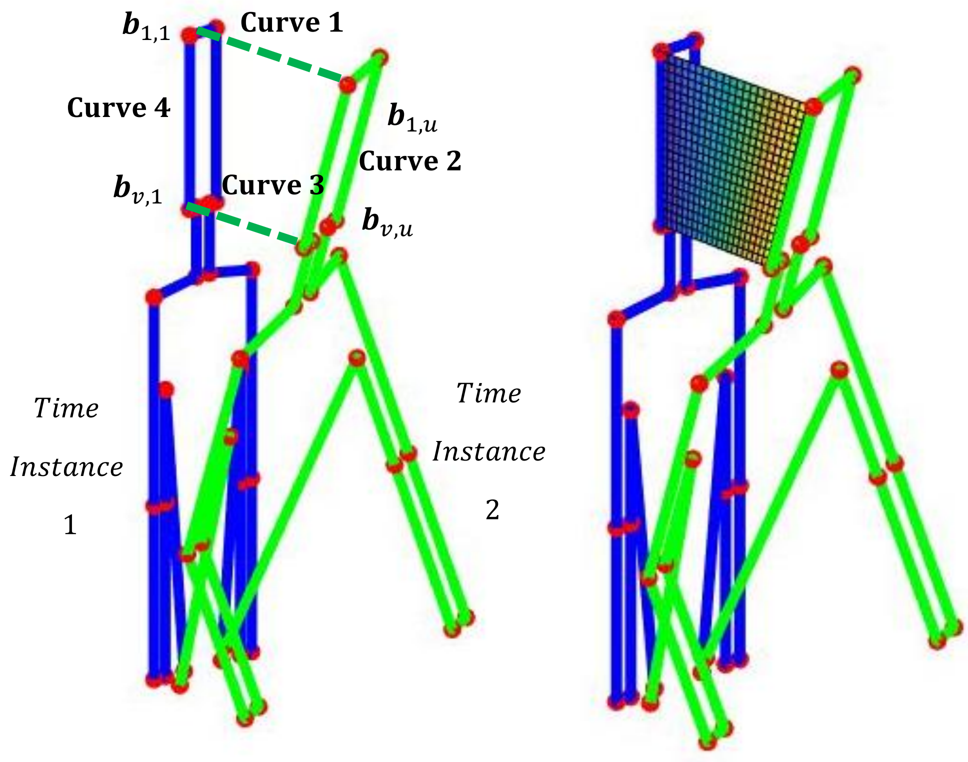


Figure 7. Definition of boundary curves and generation of a patch

With the corner points specified, the curves can be put into the proper mathematical form for application of Coons patches. The mathematical form is shown in (5), where $\boldsymbol{b}_{1,1}$ to $\boldsymbol{b}_{1,u}$ represents discrete values along Curve 1; $\boldsymbol{b}_{1,u}$ to $\boldsymbol{b}_{v,u}$ represents Curve 2; $\boldsymbol{b}_{v,u}$ to $\boldsymbol{b}_{v,1}$ represents Curve 3; and $\boldsymbol{b}_{v,1}$ to $\boldsymbol{b}_{1,1}$ represents Curve 4. Such a patch is created for each dimension. Four patches, for the X, Y, Z, and time dimensions are created. The variables *v* and *u* define the horizontal and vertical components of the mesh of the Coons patch. Higher values of *v* and *u* result in more densely packed Coons patches. Finer mesh sizes will lead a more densely defined surface, and better collision detection, but will increase computational time.

$$[\boldsymbol{b}] = \begin{bmatrix} \boldsymbol{b}_{1,1} & \cdots & \boldsymbol{b}_{1,u} \\ \vdots & \ddots & \vdots \\ \boldsymbol{b}_{v,1} & \cdots & \boldsymbol{b}_{v,u} \end{bmatrix} \tag{5}$$

The time dimension patch is also created in the same format. For the time dimension patch, curves 2 and 4 will simply be constant lines with the same time value at each point, since these curves model the human at a given time. Curves 1 and 3 are defined by linearly interpolating between the two times.

Through careful selection of *v* and *u*, computational effort can be drastically reduced without sacrificing any accuracy. This is done by selecting a minimum *v* and *u* such that the maximum distance between any two points in the Coons patch is smaller than the grid size selected at the start of the algorithm. Denser Coons patches will be mapped to the spatial grid size, so high-density patches will eventually be generalized by patches that fit the grid size in the final comparison of the swept volumes.

Since it is advantageous to uniquely determine *v* and *u* for each patch, once the boundary curves have been evaluated, a function looks at their total lengths. The *u* parameter is determined by evaluating the lengths of Curves 1 and 3. *u* is set to equal the length of the longer curve divided by half the grid size and rounding up to the nearest integer (6). Similarly, *v* is set by evaluating the lengths of Curves 2 and 4 and dividing the larger of the two by half the grid size and rounded up to the nearest integer (7). For (6-7), the $|Curve|$ parameter is simply the Euclidean distance between the two boundary points used to define the curve. In this way, it is ensured that the Coons patch can be accurately mapped to the grid size without much change in location or number of the points within the patch, and with no risk of marking as unoccupied grid locations that should be occupied. This allows each patch to have roughly the same point density regardless of how large the patch is. This is important as a small segment, like the head, should not have the same mesh as

$$u = Ceil\left(\frac{max(|Curve\ 1|, |Curve\ 3|)}{\frac{1}{2} \times Grid\ Size}\right) \tag{6}$$

$$v = Ceil\left(\frac{max(|Curve\ 2|, |Curve\ 4|)}{\frac{1}{2} \times Grid\ Size}\right) \tag{7}$$

a large segment, like the torso. Similarly, if the arm happens to be moving fast, and sweeps out a larger volume in a given amount of time than the head does, the mesh should be adjusted.

After *u* and *v* have been selected, the curves can be populated with intermediary points between the boundary points. This is done by linearly interpolating *u-2* points between the boundary points for curves 1 and 3, and *v-2* points between the boundary points for curves 2 and 4. This interpolation is completed for each dimension's patch. The interpolation for each curve at each dimension is completed by a short function that defines points at the proper spacing so that the number of points in the curve matches the mesh size. An example for curve 1 in the X dimension is shown in (8-10), where the parameter *i* steps from the second position in curve 1 to the second to last position in curve 1 (10). Completing this procedure for each dimension of each curve fully defines the boundary curves to be used in the following Coons patch equation.

$$Spacing = \frac{\boldsymbol{b}_{1,u} - \boldsymbol{b}_{1,1}}{u-1} \tag{8}$$

$$\boldsymbol{b}_{1,i} = \boldsymbol{b}_{1,1} + Spacing \times (i-1) \tag{9}$$

$$i = 2:1:(u-1) \tag{10}$$

To define the patch, let *i* and *j* represent intermediate values between the boundary curves. The points that lie at each set of indices *i* and *j* in $\boldsymbol{b}_{ij}$, where *i* and *j* range from 2 to *v-1* and 2 to *u-1* respectively, are calculated in the discrete equation of a Coons patch (11). Each patch contains a discretized grid of points. Each point represents the weighted average of the closest points on the boundary curves and the points at the boundary corners with respect to the distance between the point and each curve.

$$\boldsymbol{b}_{i,j} = \left(1 - {}^{i}/_{v}\right)\boldsymbol{b}_{1,j} + {}^{i}/_{v}\,\boldsymbol{b}_{v,j} + \left(1 - {}^{j}/_{u}\right)\boldsymbol{b}_{i,1} + {}^{j}/_{u}\,\boldsymbol{b}_{i,u} - \begin{bmatrix} 1 - {}^{i}/_{v} & {}^{i}/_{v} \end{bmatrix} \begin{bmatrix} \boldsymbol{b}_{1,1} & \boldsymbol{b}_{1,u} \\ \boldsymbol{b}_{v,1} & \boldsymbol{b}_{v,u} \end{bmatrix} \begin{bmatrix} 1 - {}^{j}/_{u} \\ {}^{j}/_{u} \end{bmatrix} \tag{11}$$

Completing the above calculation for the patch, a $(v \times u)$ set of points within the boundary curves is created. This process is completed for each dimension of interest. A patch is made for the X, Y, and Z dimensions as well as for the time dimension. Applying this process iteratively to each segment of the human's motion between each known pose defines the volume swept out by the human. In Figure 8, the positional X, Y, and Z patches are used to plot the human's swept volume, and the time patch, displayed as color, further characterizes each point.

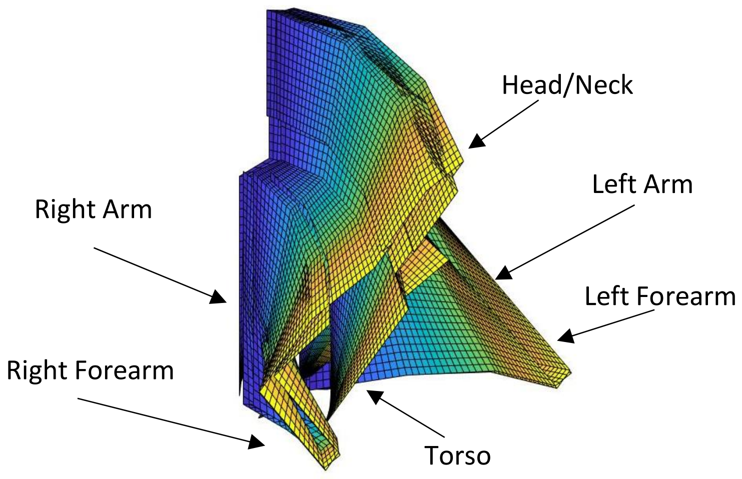


Figure 8. Iterative application of Coons patches to define the boundary surface. Time is shown as color. Darker colors represent the initial state and lighter colors represent the final state.

This iterative process can be applied to an arbitrary articulated body by supplying the algorithm with a series of sets of points that wrap around the body at various instances in time. The algorithm will go through the two such sets each iteration. It will select the first two points on the boundary curve in the first set, and the same two points in the second set. A patch will be created for this segment, and the algorithm will move on to the second and third points along the boundary curves of the first two sets. Once all the points have been used to define curves between these sets, the algorithm will repeat the process for sets two and three, and so on until all boundary points in all sets have been used. Each set of boundary points must have the same total number of points.

With the process automated, it is simple to supply the algorithm with any curve that one desires to sweep throughout time. The side profiles of the human can also be supplied to the algorithm, and the resulting surface added to the front profile to ensure that the full swept volume of the human is represented. The full human volume, along with the swept side profiles, is shown in Figure 8. By completing this procedure for the boundary curves for the robot defined previously, the swept volume shown in Figure 9 is generated.

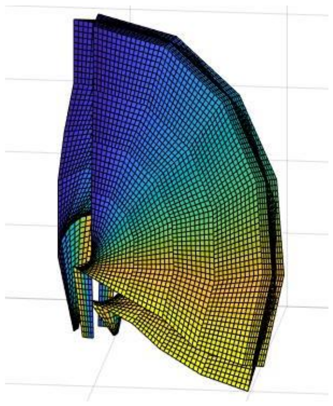


Figure 9. Iterative application of Coons patches defining robot motion. Time is shown as color.

Setting Initial and Final Conditions

The previous step defines a surface that traces the outline of the human as motion occurs. This, however, leaves the starting pose and ending pose of the human's motion unbounded by a surface. To close these ends of the surface, initial and final conditions are applied. The same formulation of Coons patches is employed but rather than patch together two different poses, patches are generated to connect all components of one pose. This requires only modification of the input data. Rather than supply boundary curves for each step-in time, the normal and orthogonal curves at one instant in time are patched together. An example of the generation of these patches for the boundary conditions is shown in figure 10. All time patches generated for the boundary conditions are populated by the time at which the object occupies that space.

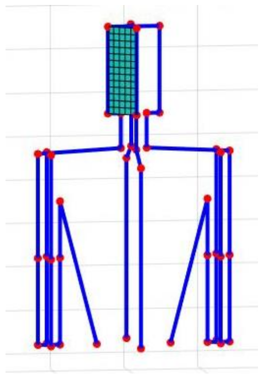


Figure 10. Definition of the boundary curves of the patches used to construct the boundary conditions at the start and end of the trajectory. Note, time is constant throughout the patch.

In the case of the head boundary condition, the normal segment of the head is connected to the corresponding orthogonal segment of the head. Then the orthogonal segment is connected to the next normal segment with a patch. This process is repeated until the head is completely wrapped. Similarly, each segment of the human is wrapped with patches. This is done for the first and the last pose. With this step, the swept volume for the human is completed (Figure 11).

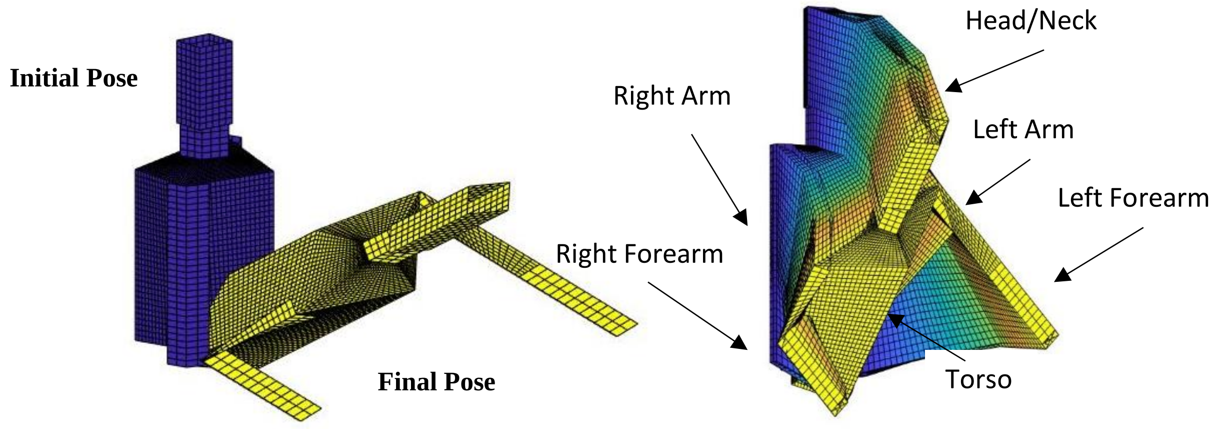


Figure 11. Start and end conditions (*Left*) defined to close composite boundary surface (*Right)*

In the same way, the initial and final conditions of the robot were modeled and overlaid on the rest of the swept volume created in the previous step. Shown below is the actual robot initial and final conditions (*Figure 11 Left*), the modeled initial and final conditions (*Figure 11 Middle*), and the composite swept volume, complete with the sweeps from the normal and orthogonal boundary curve sweeps, and the initial and final condition of the sweep (*Figure 11 Right*).

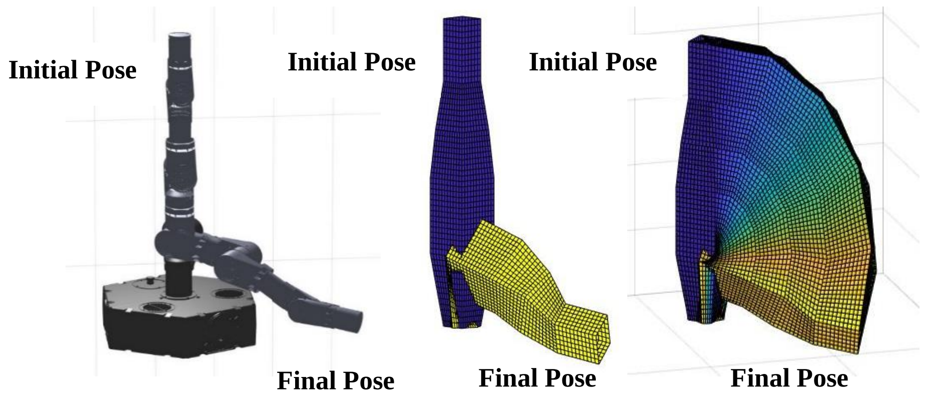


Figure 11. Model of the robot's swept volume. The start and end orientations of the robot are shown, but a total of six poses of the robot were used to generate the volume.

Finally, to check the two swept volumes for collision, the points defining each volume are mapped into a standardized grid. The grid size initially specified is used. A standard discretized time array is also used. Each point is fit to the nearest grid space, and the time associated with each point is fit to the nearest discretized time. This is done with a function that takes as input all points contained in the patches for the swept volume to be fitted cast into the form of X, Y, Z, and time arrays. This format is used to capitalize on the efficiency with which matrix operations can be performed on arrays of the same size. Each dimension is calculated in the same way. For description purposes, this process will be demonstrated for the X dimension. From the standard grid size, an upper bound, X_U, and lower bound, X_L, is generated for each point in this data set. This process is expedited by realizing that the bounds must always be consecutive multiples of the grid size. The lower bound is set by subtracting from the actual value the remainder of the quotient of the actual value and the grid size (12). The upper bound is simply found by adding the grid size to the lower bound (13).

$$X_L = X - mod(X, Grid\ Size) \tag{12}$$

$$X_U = X_L + Grid\ Size \tag{13}$$

Finally, it must be decided which bound is closest to the actual point. This is done in two steps. First, a binary array, X_{binary} is constructed by placing a 1 at all index values for which the distance between X_u and X is less than the distance between X_L and X. In this way, if X is closer to the upper bound, a 1 will be returned. All other positions are set to zero (14). Next, the closest bound is selected by setting X equal to the element by element sum of X_L and the product of the binary array and the grid size. Thus, for all positions where X is closer to the upper bound, one increment of the grid size will be added to the lower bound.

$$X_{binary} = \ (X_u - X) \ < (X - X_L) \tag{14}$$

$$X = X_L + X_{binary} \times Grid\ Size \tag{15}$$

Performing these operations on each dimension the data for the swept volume is converted to the standardized grid. This process is completed for each swept volume. Collisions occur at any location where both sweeps contain the same positional and time data; a spatiotemporal intersection of the two approximated swept volumes. An example collision is shown in Figure 12, where collision points are indicated in red.

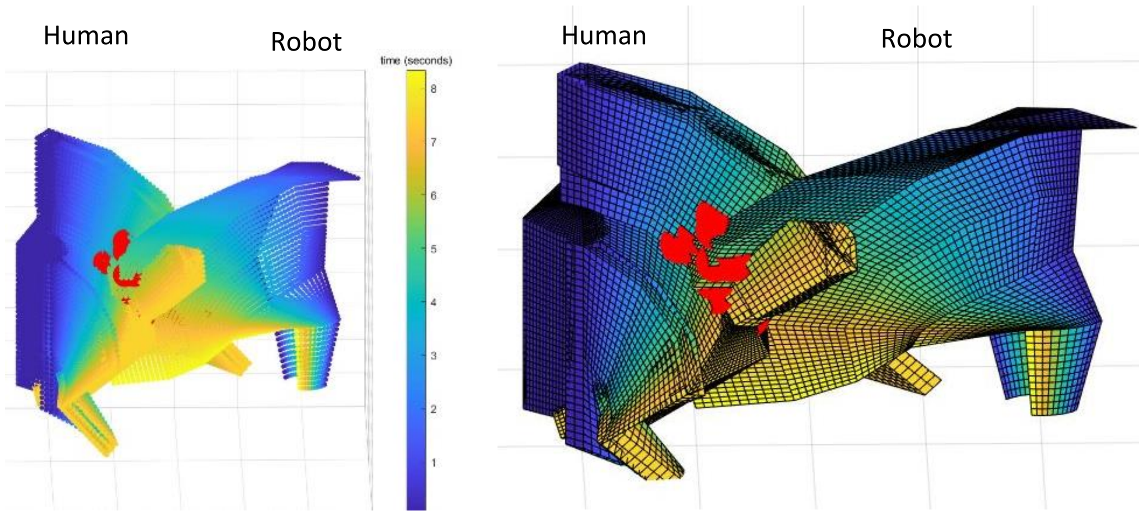


Figure 12. Collision detected between the robot and human swept volumes

Evaluation

To evaluate the algorithm, a tester was built to assign joint angle trajectories to both the human and the robot. Tests were implemented on an Intel® Core™ i7-7500 CPU processor. To evaluate computational efficiency, timed tests were run at varying grid sizes. Grid sizes between 0.01 m to 0.15 m in increments of 0.01 m were each tested. At each grid size the algorithm was run 100 times to evaluate 100 randomly generated human and robot trajectories for collision. The computational times at each grid size were averaged and plotted versus the grid size (Figure 13). Computational time versus grid size data indicate an exponential increase in computation as grid sizes decrease below 0.02 m. For grid sizes greater than 0.07 m, the algorithm runs in real time (less than 30 ms). To show in greater resolution the computational time of the algorithm at courser grid sizes, a plot displaying the data only focusing on grid sizes of 0.05 m through 0.15 m is shown in Figure 14.

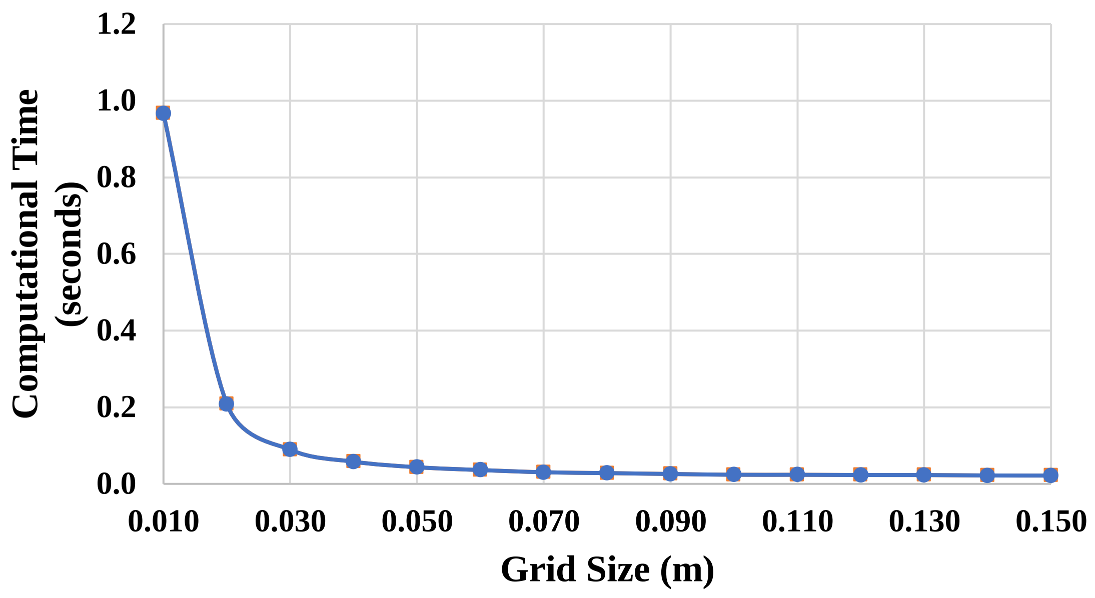


Figure 13. Computational time associated with each grid size. The trend displays monotonic behavior but exponentially for grids with a spacing of less than 2 cm.

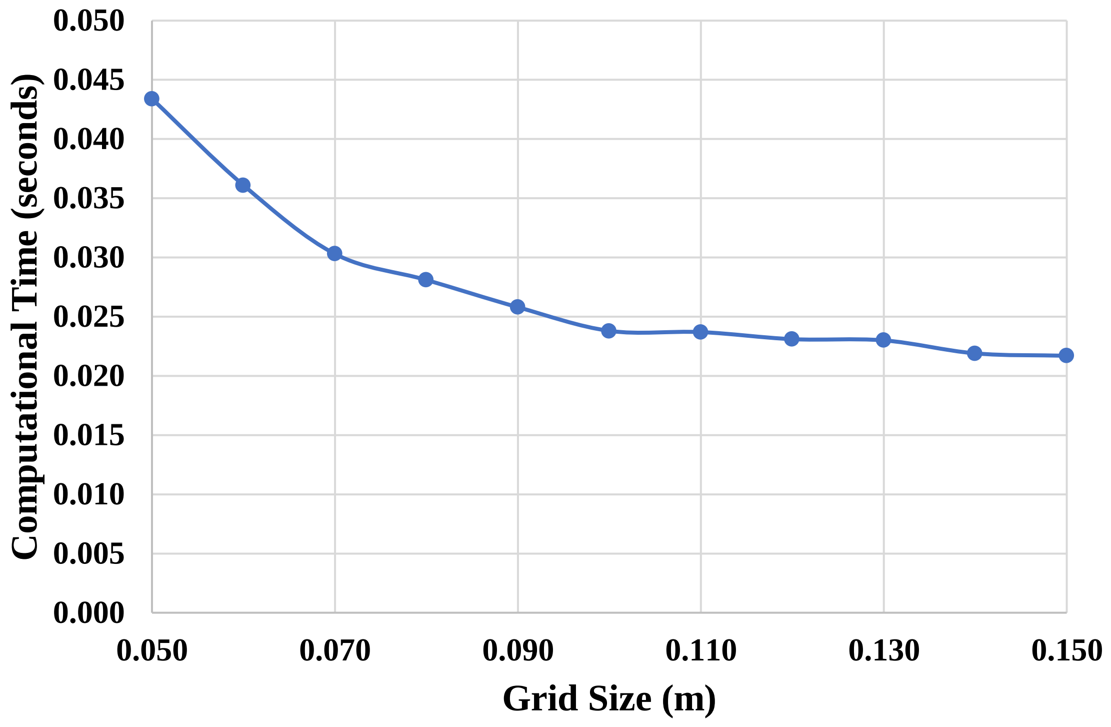


Figure 14. Computational time associated with grid sizes between 0.05 m and 0.15 m. Computational time asymptotically approaches approximately 22 ms.

The effectiveness of the algorithm at various grid sizes was evaluated for 100 test cases with randomly generated trajectories for the human and robot. Each test case was evaluated at fifteen grid sizes equally spaced between 0.01 m and 0.15 m. The goal of each test was to identify whether there was a collision or not. Of these 100 scenarios, 87 tests maintained the same prediction for all grid sizes. For 9 tests, lower resolution iterations misidentified a collision, but converged to a consistent solution for higher resolution grid sizes. The highest resolution at which the algorithm misidentified a collision was at 0.060 m, while the average grid size at which the algorithm converged was 0.110 m. There were four cases in which the algorithm did not to converge to a solution. In these 4 cases, as well as the previously discussed 8 cases, both swept volumes came indistinguishably close to each other, and collisions could not even be determined upon visual inspection. These results indicate that the algorithm is successful in identifying collisions for grid sizes less than 0.06 m. Thus, algorithm can be expected to perform accurate collision detection for grid sizes of 0.07 m and larger in less than 30 ms.

Cases in which the algorithm is inconsistent in identifying a collision are of concern. While on one hand, this means that the robot will potentially become over cautious, identifying collisions that never actually happen, it can also cause the robot to miss a collision and proceed unsafely. Since safety is the utmost priority when it comes to making the decision between safety and productivity, two aspects of the algorithm have been designed to play in the favor of safety.

The first aspect is, of course, the safety factor. This factor, in effect, exaggerates the modeled human so that fringe cases of the algorithm, in which collisions are inconsistently identified, occur well outside the actual bounds of the human. The rate of expansion can further be tuned. The algorithm is set up in such a way that this parameter is easy to adjust according to the situation. For situations in which safety is of paramount importance, such as in the case of human collaborators with low confidence in the robot or operations with high payloads or dangerous tools, this factor can be drastically increased. To validate this claim, one of the fringe cases for which the algorithm had trouble identifying the collision was investigated. This case was rerun with a greater safety factor. It was found that by increasing the safety factor, the algorithm was able to identify collisions at all grid sizes.

Secondly, the algorithm is designed to operate in real time. From the above discussion it was shown that the algorithm can reliably run in less than 30 ms. This means that if the algorithm misses a collision, it will have another opportunity to identify the collision shortly thereafter. This drastically increases the chances of the algorithm detecting a collision, and since a false collision detection is preferable to a missed collision, this result is satisfactory.

Computational Effect of Collision Detection with Multiple Humans

To further investigate the effects of environment modeling on computational time, more human models were introduced to the environment. The same testing procedure as the one previously conducted to evaluate computational efficiency was implemented for varying numbers of humans modeled in the environment with the robot. In addition to the results from the previous test, four different scenarios were evaluated at grid sizes varying between 0.01 m and 0.15 m in increments of 0.01 m. In scenario one, two humans were introduced to the environment. Scenario two contained three humans, scenario three contained four humans, and scenario four contained five humans. Figure 15 depicts the second scenario.

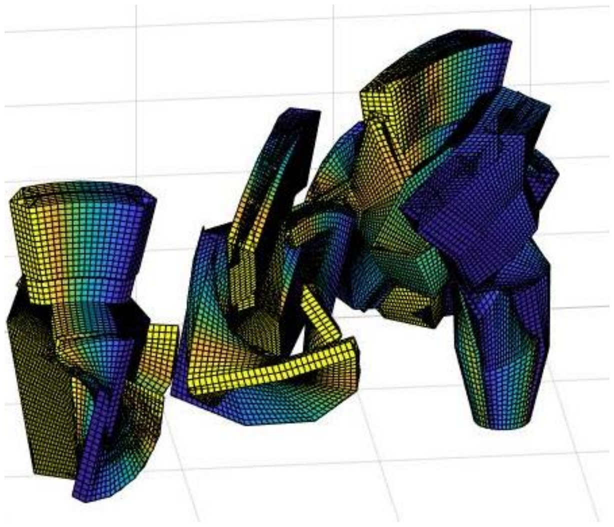


Figure 15. Workspace constructed with a robot, positioned in the front left of the workspace, along with three humans.

For each scenario and at each grid size, 100 timed tests were run. As before, each test implemented the full algorithm. Random joint trajectories for all humans and the robot were generated for each run. The run times for the 100 tests were averaged and can be seen in Figure 16 plotted against the corresponding grid size.

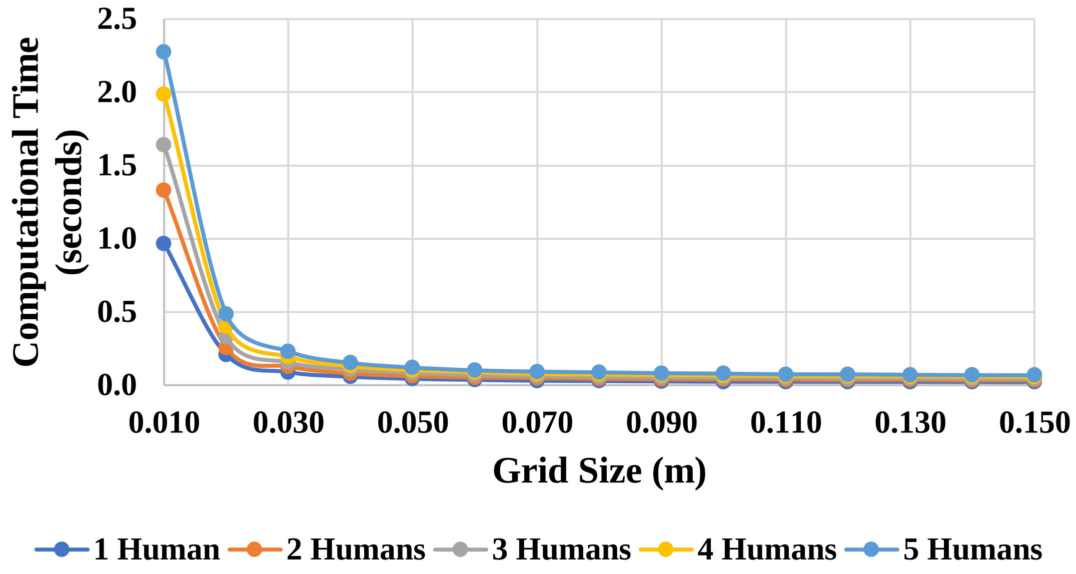


Figure 16. Computational time associated with each grid size for scenarios involving varying numbers of humans in the environment. As seen before, at courser grid sizes, the algorithm converges to a value but behaves exponentially for grids with a spacing of less than 2 cm.

Figure 16 contains the performance of the algorithm for each scenario in addition to the results from the previous computational testing for reference. To show in greater resolution the computational time of the algorithm at courser grid sizes for each scenario, a plot displaying the same data only focusing on grid sizes of 0.05 m through 0.15 m is shown in Figure 17.

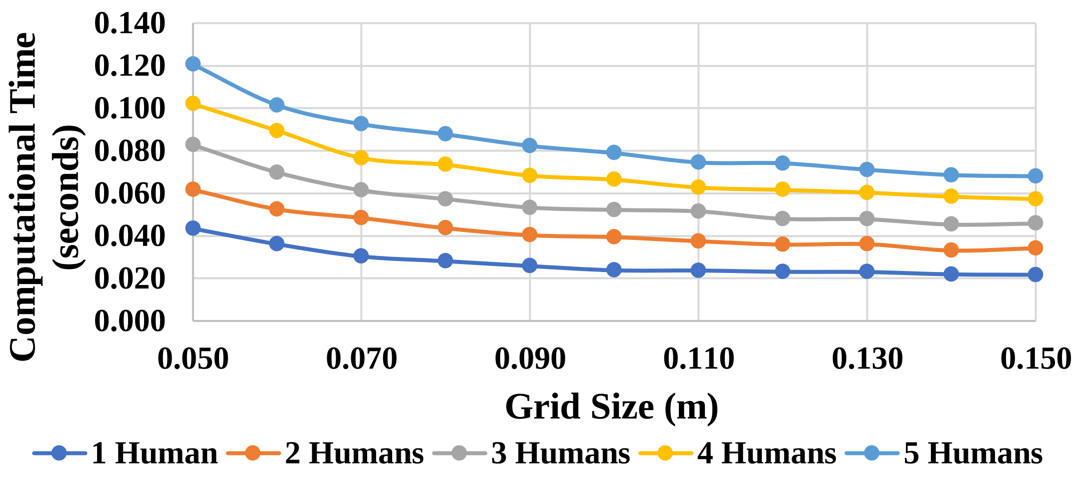


Figure 17. Computational time associated with grid sizes between 0.05 m and 0.15 m for each scenario. In all scenarios computational time asymptotically approaches a value.

From the plots alone, it can be seen that the addition of humans to be modeled in the environment seems to have a proportional effect on the computational time. To validate this observation further, the relationship between the average computational time of the initial test, called from here on the control, and the average computational time of each scenario will be investigated. The average computational times are displayed in Table I. Additionally the ratio of the average computational times of each scenario to the control is also shown.

TABLE I

COMPARISON CONTROL OF RUN TIME DATA FOR SCENARIOS WITH MULTIPLE OBJECTS

	Control	Scenario 1		Scenario 2		Scenario 3		Scenario 4	
Grid Size (m)	Average Run Time (seconds)	Average Run Time (seconds)	Ratio of Scenario One to Control	Average Run Time (seconds)	Ratio of Scenario Two to Control	Average Run Time (seconds)	Ratio of Scenario Three to Control	Average Run Time (seconds)	Ratio of Scenario Four to Control
0.01	0.967	1.330	1.38	1.639	1.70	1.987	2.06	2.275	2.35
0.02	0.208	0.257	1.23	0.330	1.58	0.403	1.93	0.484	2.32
0.03	0.089	0.127	1.42	0.157	1.75	0.194	2.17	0.231	2.59
0.04	0.058	0.081	1.39	0.109	1.89	0.130	2.25	0.152	2.63
0.05	0.043	0.062	1.42	0.083	1.91	0.102	2.35	0.121	2.78
0.06	0.036	0.053	1.45	0.070	1.93	0.089	2.48	0.101	2.81
0.07	0.030	0.048	1.60	0.061	2.03	0.077	2.53	0.093	3.05
0.08	0.028	0.044	1.55	0.057	2.04	0.073	2.61	0.088	3.12
0.09	0.026	0.040	1.56	0.053	2.07	0.068	2.65	0.082	3.19
0.1	0.024	0.039	1.66	0.052	2.19	0.066	2.79	0.079	3.32
0.11	0.024	0.038	1.58	0.052	2.17	0.063	2.65	0.075	3.14
0.12	0.023	0.036	1.55	0.048	2.08	0.062	2.67	0.074	3.21
0.13	0.023	0.036	1.57	0.048	2.08	0.060	2.62	0.071	3.09
0.14	0.022	0.033	1.51	0.045	2.07	0.058	2.67	0.069	3.13
0.15	0.022	0.034	1.58	0.046	2.11	0.057	2.64	0.068	3.13

It should be noted that the calculated ratios remain fairly consistent for each scenario across all tested grid sizes. This is especially the case for courser grid sizes, where the relationship between the grid size and the computational time is closer to linear. These ratios indicate the proportional nature of the increase in computational time with respect to the number of humans modeled. In scenario one, after doubling the number of humans modeled in the environment, the computational time increases by only a factor of roughly 1.5. In scenario two, after tripling the number of humans modeled in the environment, the computational time increases by a factor of roughly 2. For scenarios three and four this factor becomes approximately 2.5 and 3. A diminish in the additional amount of computational time for each addition of another modeled object to the environment is to be expected in that the initial overhead of the algorithm, including amongst other things the modeling of the robot, is not increased. The computational addition is only of a section, a large section, of the algorithm. Thus, it is expected, and verified for a few cases, that as objects are added to the environment, the addition in computational time should diminish. As many more objects are added to the environment, however, the additional computational time should become constant. In general, for all additions of objects to the environment, the addition in computational time will be less than the product of the computational time before the addition and the ratio of the number of objects after the addition to the number of objects before the addition.

Conclusion

The contribution of this thesis is a quick, effective predictive collision detection algorithm designed to run in real time as a background operation of a robot controller. The novelty of the approach was to use Coons patches to interpolate between sampled orientations of a body throughout time. In this way, continuous collision detection is implemented with minimal computation. This is a building block to a larger effort to develop HRC techniques. Through various tests, the algorithm was shown to run in real time on the order of 30 ms. Furthermore, the algorithm exhibited a high degree of effectiveness in identifying collisions.

This algorithm was developed to function in tandem with the collaborative work between multiple universities and a corporate sponsor. As such, it was designed to function on the predicted data from both a team working on a sensor suite and a team working on prediction of the environment and the human behavior. This data is expected to predict the human’s orientation throughout time up to three seconds in advance. Thus, in this paper, it was assumed that only a few orientations throughout time of the human and robot were needed to construct the sweep. If prediction capabilities of the human were extended, more orientations throughout time would be available and required for an accurate model of the sweep. Additionally, since the patches linearly interpolate between poses, for trajectories that exhibit a high degree of curvature, more poses are needed for accurate modeling. Each time a pose is added, an additional set of patches must be generated, increasing computational time. A suggested extension of this work would be to study the computational effects of increasing the number of poses. Furthermore, it would be valuable to develop an evaluation of the curvature of trajectories. This ranking must be a quick check that can be performed on the data before execution of the algorithm. The goal of determining the curvature would be to optimally set the required number of orientations before running the algorithm.

Additional future works will include optimizing the grid size according to the input trajectory to accurately model necessary features with the largest grid size possible and implementing the algorithm with sensor data recording interaction between a physical human and the robot to evaluate the algorithm with hardware in the loop testing. Additionally, due to the discretization of the data, there potentially exists the chance that a collision can be missed because two surfaces can skip past each other. Further work will evaluate pertinent parameters to this phenomenon, such as size and speed of the obstacle, and develop guidelines to avoid this by properly setting the mesh and grid size. Finally, this algorithm is to be used as a tool in a suite of tools designed for HRC. Once this algorithm identifies a collision, the next step is to optimally avoid the collision. The next major milestone in this research will be to develop such a capability for the robot. Due to the high frequency application of this algorithm during operation, this collision avoidance algorithm must be designed in a way that will avoid dithering when fringe cases falter between false detections and misses. To address this issue, the focus of the collision avoidance algorithm will be on capitalizing on the trajectory of the robot known a priori. Rather than entirely re-planning the trajectory, this algorithm will attempt to minimally modify the current trajectory such that it is moved out of the way just enough to miss the human. In this way, the effects of dithering will be less drastic as the fringe cases will only involve slight changes in the trajectory. The algorithm developed in this work is thus a necessary first step in a series of algorithms designed to enhance robot performance in human robot collaboration.

Acknowledgements

Funding was provided by University of Florida’s Scholars Program. Research conducted in support of NSF/NRI: INT: COLLAB: Manufacturing USA: Intelligent Human-Robot Collaboration for Smart Factory (Award I.D. #:1830383). . Any opinions, findings and conclusions or recommendations expressed are those of the researchers and do not necessarily reflect the views of the National Science Foundation.

Appendix A

Rotation Matrices:

*These are a function of input joints for the angle of rotation α

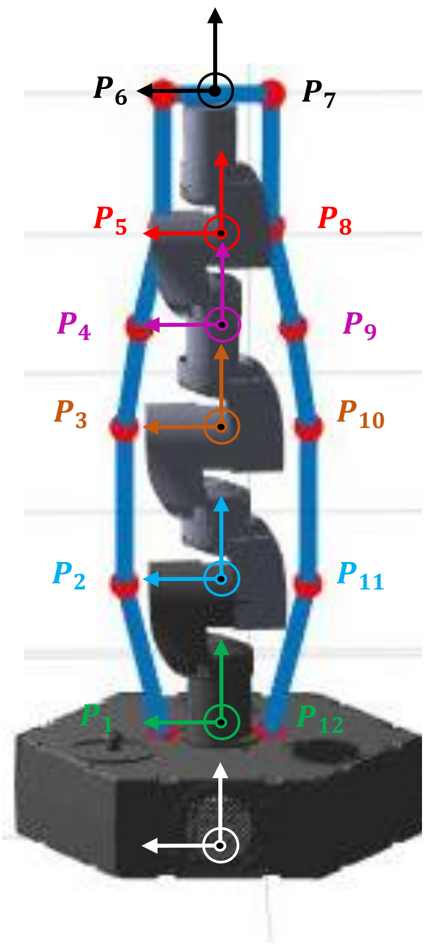
*These are applicable to any joint as all systems are defined such that they rotate about their X, Y, or Z axes

$$R_x = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & cos(\alpha) & -sin(\alpha) & 0 \\ 0 & sin(\alpha) & cos(\alpha) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$R_y = \begin{bmatrix} cos(\alpha) & 0 & sin(\alpha) & 0 \\ 0 & 1 & 0 & 0 \\ -sin(\alpha) & 0 & cos(\alpha) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$R_z = \begin{bmatrix} cos(\alpha) & -sin(\alpha) & 0 \\ sin(\alpha) & cos(\alpha) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Fixed System	
System 1	🟩
System 2	🟦
System 3	🟫
System 4	🟪
System 5	🟥
System 6	⬛



Transformation Matrices:

*These are a function of robot geometry

$$ {}^{f}_{1}T = \begin{bmatrix} 1 & 0 & 0 & 0.06103 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1.135 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$ {}^{1}_{2}T = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0.202 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$ {}^{2}_{3}T = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0.2105 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$ {}^{3}_{4}T = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0.134 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$ {}^{4}_{5}T = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0.134 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$ {}^{5}_{6}T = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0.1745 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Equations for Location of Each Boundary Point in the Fixed System

$${}^{f}P_1 = {}^{f}_{1}T\ R_1\ {}^{1}P_1$$

$${}^{f}P_{12} = {}^{f}_{1}T\ R_1\ {}^{1}P_{12}$$

Where R_1 corresponds to a rotation of α_1 in R_z, ${}^{1}P_1 = [0.07\ 0\ 0\ 1]'$ and ${}^{1}P_{12} = [-0.07\ 0\ 0\ 1]'$

$${}^{f}P_2 = {}^{f}_{1}T\ R_1\ {}^{1}_{2}T\ R_2\ {}^{2}P_2$$

$${}^{f}P_{11} = {}^{f}_{1}T\ R_1\ {}^{1}_{2}T\ R_2\ {}^{2}P_{11}$$

Where R_2 corresponds to a rotation of α_2 in R_x, ${}^{2}P_2 = [0.12\ 0\ 0\ 1]'$ and ${}^{2}P_{11} = [-0.12\ 0\ 0\ 1]'$

$${}^{f}P_3 = {}^{f}_{1}T\ R_1\ {}^{1}_{2}T\ R_2\ {}^{2}_{3}T\ R_3\ {}^{3}P_3$$

$${}^{f}P_{10} = {}^{f}_{1}T\ R_1\ {}^{1}_{2}T\ R_2\ {}^{2}_{3}T\ R_3\ {}^{3}P_{10}$$

Where R_3 corresponds to a rotation of α_3 in R_x, ${}^{3}P_3 = [0.12\ 0\ 0\ 1]'$ and ${}^{3}P_{10} = [-0.12\ 0\ 0\ 1]'$

$${}^{f}P_4 = {}^{f}_{1}T\ R_1\ {}^{1}_{2}T\ R_2\ {}^{2}_{3}T\ R_3\ {}^{3}_{4}T\ R_4\ {}^{4}P_4$$

$${}^{f}P_9 = {}^{f}_{1}T\ R_1\ {}^{1}_{2}T\ R_2\ {}^{2}_{3}T\ R_3\ {}^{3}_{4}T\ R_4\ {}^{4}P_9$$

Where R_4 corresponds to a rotation of α_4 in R_z, ${}^{4}P_4 = [0.1\ 0\ 0\ 1]'$ and ${}^{4}P_9 = [-0.1\ 0\ 0\ 1]'$

$${}^{f}P_5 = {}^{f}_{1}T\ R_1\ {}^{1}_{2}T\ R_2\ {}^{2}_{3}T\ R_3\ {}^{3}_{4}T\ R_4\ {}^{4}_{5}T\ R_5\ {}^{5}P_5$$

$${}^{f}P_8 = {}^{f}_{1}T\ R_1\ {}^{1}_{2}T\ R_2\ {}^{2}_{3}T\ R_3\ {}^{3}_{4}T\ R_4\ {}^{4}_{5}T\ R_5\ {}^{5}P_8$$

Where R_5 corresponds to a rotation of α_5 in R_x, ${}^{5}P_5 = [0.07\ 0\ 0\ 1]'$ and ${}^{5}P_8 = [-0.07\ 0\ 0\ 1]'$

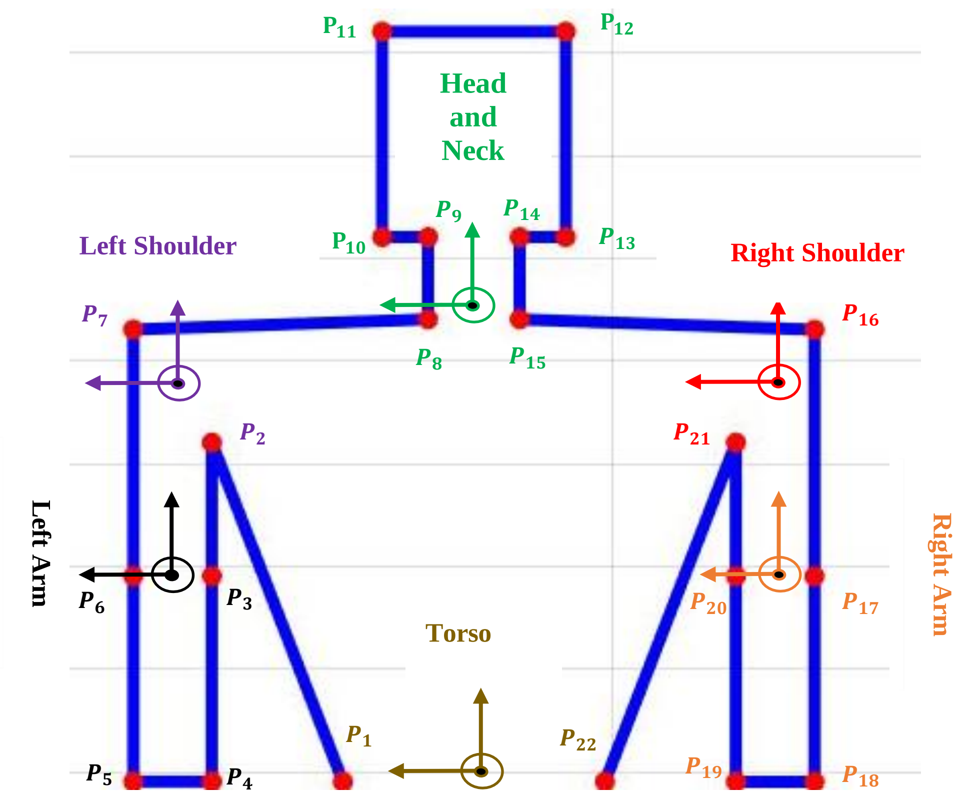
$${}^{f}P_6 = {}^{f}_{1}T\ R_1\ {}^{1}_{2}T\ R_2\ {}^{2}_{3}T\ R_3\ {}^{3}_{4}T\ R_4\ {}^{4}_{5}T\ R_5\ {}^{5}_{6}T\ R_6\ {}^{6}P_6$$

$${}^{f}P_7 = {}^{f}_{1}T\ R_1\ {}^{1}_{2}T\ R_2\ {}^{2}_{3}T\ R_3\ {}^{3}_{4}T\ R_4\ {}^{4}_{5}T\ R_5\ {}^{5}_{6}T\ R_6\ {}^{6}P_7$$

Where R_6 corresponds to a rotation of α_6 in R_z, ${}^{6}P_6 = [0.07\ 0\ 0\ 1]'$ and ${}^{6}P_7 = [-0.07\ 0\ 0\ 1]'$

*To determine the location of the points in the orthogonal boundary curves, simple swap the values in the X and Y positions of the array for each point.

Appendix B



Fixed System	
System 1	🟫
System 2	🟪
System 3	⬛
System 4	🟥
System 5	🟧
System 6	🟩

Transformation Matrices:

***These are a function of robot geometry**

$$ {}^{f}_{1}T = \begin{bmatrix} 1 & 0 & 0 & -0.1 \\ 0 & 1 & 0 & -0.8 \\ 0 & 0 & 1 & 0.1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad {}^{1}_{2}T = \begin{bmatrix} 1 & 0 & 0 & 0.23 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0.4 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad {}^{2}_{3}T = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & -0.2 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

$$ {}^{1}_{4}T = \begin{bmatrix} 1 & 0 & 0 & -0.23 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0.4 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad {}^{4}_{5}T = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & -0.2 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad {}^{1}_{6}T = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0.45 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

Rotation Matrices:

***These are a function of input joints for the angle of rotation α**

*These are applicable to any joint as all systems are defined such that they rotate about their X, Y, or Z axes

$$ R_x = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & cos(\alpha) & -sin(\alpha) & 0 \\ 0 & sin(\alpha) & cos(\alpha) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad R_y = \begin{bmatrix} cos(\alpha) & 0 & sin(\alpha) & 0 \\ 0 & 1 & 0 & 0 \\ -sin(\alpha) & 0 & cos(\alpha) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad R_z = \begin{bmatrix} cos(\alpha) & -sin(\alpha) & 0 & 0 \\ sin(\alpha) & cos(\alpha) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

Definition of the Safety Factors

For each iteration, K increases. K is defined as the product of the change in time between the current orientation and the initial orientation, ΔT, and a scaling factor, n, selected by the user. This scaling factor gives the user control over the safety threshold used.

Before implementation of the following kinematics, the following safety factors are defined for each body part of the human. In this implementation, all factors expand at the same rate. A simple adjustment would allow for faster or slower expansion rates depending on the body part. Simply multiply the value $K + 1$ by the desired weighting factor.

$$K = \Delta T \times n$$

Body Part	**Variable Name**	**Value**
Head	SF_H	$K + 1$
Neck	SF_N	$K + 1$
Left Shoulder	SF_LS	$K + 1$
Right Shoulder	SF_RS	$K + 1$
Left Arm	SF_LA	$K + 1$
Right Arm	SF_RA	$K + 1$
Left Forearm	SF_LF	$K + 1$
Right Forearm	SF_RF	$K + 1$
Torso	SF_T	$K + 1$
Waist	SF_W	$K + 1$

Equations for Location of Each Boundary Point in the Fixed System

Torso Points

$$^{f}P_1 = {}^{f}_{1}T\ R_{1x}R_{1y}R_{1z}\ {}^{1}P_1$$

$$^{f}P_{22} = {}^{f}_{1}T\ R_{1x}R_{1y}R_{1z}\ {}^{1}P_{22}$$

Where R_{1x} corresponds to a rotation of α_{1x} about system 1's X axis, R_{1y} corresponds to a rotation of α_{1y} about system 1's Y axis, and R_{1z} corresponds to a rotation of α_{1z} about system 1's Z axis. $^{1}P_1 = [0.1 * SF\_T\ 0\ 0\ 1]'$ and $^{1}P_{22} = [-0.1 * SF\_T\ 0\ 0\ 1]'$

Left Shoulder Points

$$^{f}P_2 = {}^{f}_{1}T\ R_{1x}R_{1y}R_{1z}\ {}^{1}_{2}T\ R_{2x}R_{2y}R_{2z}\,{}^{2}P_2$$

$$^{f}P_7 = {}^{f}_{1}T\ R_{1x}R_{1y}R_{1z}\ {}^{1}_{2}T\ R_{2x}R_{2y}R_{2z}\,{}^{2}P_7$$

Where R_{2x} corresponds to a rotation of α_{2x} about system 2's X axis, R_{2y} corresponds to a rotation of α_{2y} about system 2's Y axis, and R_{2z} corresponds to a rotation of α_{2z} about system 2's Z axis. $^{2}P_2 = [-0.03 * SF\ 0\ -0.07\ 1]'$,$^{2}P_7 = [0.03 * SF\ 0\ 0.04 + 0.03(SF\_H - 1)\ 1]'$, and $SF = max(SF\_T, SF\_LS)$

Left Arm Points

$$^{f}P_3 = {}^{f}_{1}T\ R_{1x}R_{1y}R_{1z}\ {}^{1}_{2}T\ R_{2x}R_{2y}R_{2z}\ {}^{2}_{3}T\ R_{3x}\,{}^{3}P_3$$

$$^{f}P_6 = {}^{f}_{1}T\ R_{1x}R_{1y}R_{1z}\ {}^{1}_{2}T\ R_{2x}R_{2y}R_{2z}\ {}^{2}_{3}T\ R_{3x}\,{}^{3}P_6$$

$$^{f}P_4 = {}^{f}_{1}T\ R_{1x}R_{1y}R_{1z}\ {}^{1}_{2}T\ R_{2x}R_{2y}R_{2z}\ {}^{2}_{3}T\ R_{3x}\,{}^{3}P_4$$

$$^{f}P_5 = {}^{f}_{1}T\ R_{1x}R_{1y}R_{1z}\ {}^{1}_{2}T\ R_{2x}R_{2y}R_{2z}\ {}^{2}_{3}T\ R_{3x}\,{}^{3}P_5$$

R_{3x} corresponds to a rotation of α_{3x} about system 3's X axis. $^{3}P_3 = [-0.03 * SF\ 0\ 0\ 1]'$, $^{3}P_6 = [0.03 * SF\ 0\ 0\ 1]'$, $^{3}P_4 = [-0.03 * SF\_LF\ 0\ -0.2\ 1]'$, and $^{3}P_5 = [0.03 * SF\_LF\ 0\ -0.2\ 1]'$, where $SF = max(SF\_LF, SF\_LA)$.

Right Shoulder Points

$${}^{f}P_{21} = {}^{f}_{1}T\ R_{1x}R_{1y}R_{1z}\ {}^{1}_{4}T\ R_{4x}R_{4y}R_{4z}\,{}^{4}P_{21}$$

$${}^{f}P_{16} = {}^{f}_{1}T\ R_{1x}R_{1y}R_{1z}\ {}^{1}_{4}T\ R_{4x}R_{4y}R_{4z}\,{}^{4}P_{16}$$

Where R_{4x} corresponds to a rotation of α_{4x} about system 4's X axis, R_{4y} corresponds to a rotation of α_{4y} about system 4's Y axis, and R_{4z} corresponds to a rotation of α_{4z} about system 4's Z axis. Additionally, ${}^{4}P_{21} = [0.03 * SF\ 0\ \ -0.07\ 1]'$ and ${}^{4}P_{16} = [-0.03 * SF\ 0\ 0.04 + 0.03(SF\_H - 1)\ 1]'$, and $SF\ =\ max(SF\_T, SF\_RS)$.

Right Arm Points

$${}^{f}P_{17} = {}^{f}_{1}T\ R_{1x}R_{1y}R_{1z}{}^{1}_{4}T\ R_{4x}R_{4y}R_{4z}\ {}^{4}_{5}T\ R_{5x}\,{}^{5}P_{17}$$

$${}^{f}P_{20} = {}^{f}_{1}T\ R_{1x}R_{1y}R_{1z}\ {}^{1}_{4}T\ R_{4x}R_{4y}R_{4z}\ {}^{4}_{5}T\ R_{5x}\,{}^{5}P_{20}$$

$${}^{f}P_{18} = {}^{f}_{1}T\ R_{1x}R_{1y}R_{1z}\ {}^{1}_{4}T\ R_{4x}R_{4y}R_{4z}\ {}^{4}_{5}T\ R_{5x}\,{}^{5}P_{18}$$

$${}^{f}P_{19} = {}^{f}_{1}T\ R_{1x}R_{1y}R_{1z}\ {}^{1}_{4}T\ R_{4x}R_{4y}R_{4z}\ {}^{4}_{5}T\ R_{5x}\,{}^{5}P_{19}$$

R_{5x} corresponds to a rotation of α_{5x} about system 3's X axis. ${}^{5}P_{17} = [-0.03 * SF\ 0\ 0\ 1]'$, ${}^{5}P_{20} = [0.03 * SF\ 0\ 0\ 1]'$, ${}^{5}P_{18} = [-0.03 * SF\_RF\ 0\ \ -0.02\ 1]'$, and ${}^{5}P_{19} = [0.03 * SF\_RF\ 0\ \ -0.02\ 1]'$, where $SF\ =\ max(SF\_RF, SF\_RA)$.

Head and Neck Points

$${}^{f}P_{8} = {}^{f}_{1}T\ R_{1x}R_{1y}R_{1z}\ {}^{1}_{6}T\ R_{6x}R_{6y}R_{6z}\,{}^{6}P_{8}$$

$${}^{f}P_{15} = {}^{f}_{1}T\ R_{1x}R_{1y}R_{1z}\ {}^{1}_{6}T\ R_{6x}R_{6y}R_{6z}\,{}^{6}P_{15}$$

$${}^{f}P_{9} = {}^{f}_{1}T\ R_{1x}R_{1y}R_{1z}\ {}^{1}_{6}T\ R_{6x}R_{6y}R_{6z}\,{}^{6}P_{9}$$

$${}^{f}P_{14} = {}^{f}_{1}T\ R_{1x}R_{1y}R_{1z}{}^{1}_{6}T\ R_{6x}R_{6y}R_{6z}\,{}^{6}P_{14}$$

$${}^{f}P_{10} = {}^{f}_{1}T\ R_{1x}R_{1y}R_{1z}\ {}^{1}_{6}T\ R_{6x}R_{6y}R_{6z}\,{}^{6}P_{10}$$

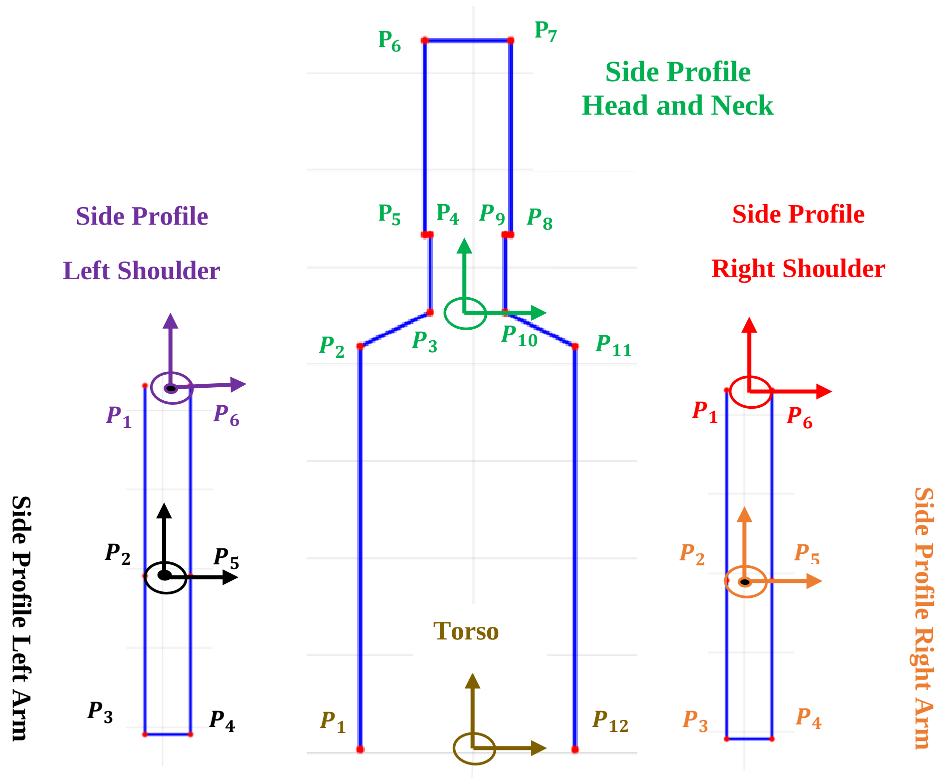
$${}^{f}P_{13} = {}^{f}_{1}T\ R_{1x}R_{1y}R_{1z}\ {}^{1}_{6}T\ R_{6x}R_{6y}R_{6z}\,{}^{6}P_{13}$$

$${}^{f}P_{11} = {}^{f}_{1}T\ R_{1x}R_{1y}R_{1z}\ {}^{1}_{6}T\ R_{6x}R_{6y}R_{6z}\,{}^{6}P_{11}$$

$${}^{f}P_{12} = {}^{f}_{1}T\ R_{1x}R_{1y}R_{1z}\ {}^{1}_{6}T\ R_{6x}R_{6y}R_{6z}\,{}^{6}P_{12}$$

Where R_{6x} corresponds to a rotation of α_{6x} about system 6's X axis, R_{6y} corresponds to a rotation of α_{6y} about system 6's Y axis, and R_{6z} corresponds to a rotation of α_{6z} about system 6's Z axis. Additionally, ${}^{6}P_{8} = [0.035 * SF\_N\ 0\ 0\ 1]'$, ${}^{6}P_{15} = [-0.035 * SF\_N\ 0\ 0\ 1]'$, ${}^{6}P_{9} = [0.035 * SF\_N\ 0\ 0.08\ 1]'$, ${}^{6}P_{14} = [-0.035 * SF\_N\ 0\ 0.08\ 1]'$, ${}^{6}P_{10} = [0.07 * SF\_H\ 0\ 0.08\ 1]'$, ${}^{6}P_{13} = [-0.07 * SF\_H\ 0\ 0.08\ 1]'$, ${}^{6}P_{11} = [0.07 * SF\_H\ 0\ 0.28 + 0.07(SF\_H - 1)\ 1]'$, ${}^{6}P_{12} = [-0.07 * SF\_H\ 0\ 0.28 + 0.07(SF\_H - 1)\ 1]'$,

Appendix C



Fixed System	⬜
System 1	🟨
System 2	🟪
System 3	⬛
System 4	🟥
System 5	🟧
System 6	🟩

The side profiles utilize the same coordinate systems as defined in Appendix B.

Transformation and rotation matrices for each joint are the same as defined in Appendix B.

All safety factors used are the same as those outlined in Appendix B.

Equations for Location of the Three Side Profile Boundary Points in the Fixed System

Left Shoulder Points

$$ {}^{f}P_1 = {}^{f}_{1}T\ R_{1x}R_{1y}R_{1z}\ {}^{1}_{2}T\ R_{2x}R_{2y}R_{2z}\,{}^{2}P_1 $$

$$ {}^{f}P_6 = {}^{f}_{1}T\ R_{1x}R_{1y}R_{1z}\ {}^{1}_{2}T\ R_{2x}R_{2y}R_{2z}\,{}^{2}P_6 $$

Where R_{1x} corresponds to a rotation of α_{1x} about system 1's X axis, R_{1y} corresponds to a rotation of α_{1y} about system 1's Y axis, and R_{1z} corresponds to a rotation of α_{1z} about system 1's Z axis. R_{2x} corresponds to a rotation of α_{2x} about system 2's X axis, R_{2y} corresponds to a rotation of α_{2y} about system 2's Y axis, and R_{2z} corresponds to a rotation of α_{2z} about system 2's Z axis. ${}^{2}P_1 = [0\ -0.03 * SF\ -0.07\ 1]'$,${}^{2}P_6 = [0\ 0.03 * SF\ 0.04 + 0.03(SF\_H - 1)\ \ 1]'$, and $SF = max(SF\_T, SF\_LS)$

Left Arm Points

$$ {}^{f}P_2 = {}^{f}_{1}T\ R_{1x}R_{1y}R_{1z}\ {}^{1}_{2}T\ R_{2x}R_{2y}R_{2z}\ {}^{2}_{3}T\ R_{3x}\,{}^{3}P_2 $$

$$ {}^{f}P_5 = {}^{f}_{1}T\ R_{1x}R_{1y}R_{1z}\ {}^{1}_{2}T\ R_{2x}R_{2y}R_{2z}\ {}^{2}_{3}T\ R_{3x}\,{}^{3}P_5 $$

$$ {}^{f}P_3 = {}^{f}_{1}T\ R_{1x}R_{1y}R_{1z}\ {}^{1}_{2}T\ R_{2x}R_{2y}R_{2z}\ {}^{2}_{3}T\ R_{3x}\,{}^{3}P_3 $$

$$ {}^{f}P_4 = {}^{f}_{1}T\ R_{1x}R_{1y}R_{1z}\ {}^{1}_{2}T\ R_{2x}R_{2y}R_{2z}\ {}^{2}_{3}T\ R_{3x}\,{}^{3}P_4 $$

R_{3x} corresponds to a rotation of α_{3x} about system 3's X axis. ${}^{3}P_2 = [0\ -0.03 * SF\ 0\ 1]'$, ${}^{3}P_5 = [0\ 0.03 * SF\ 0\ 1]'$, ${}^{3}P_3 = [0\ -0.03 * SF\_LF\ -0.2\ 1]'$, and ${}^{3}P_4 = [0\ 0.03 * SF\_LF\ -0.2\ 1]'$, where $SF = max(SF\_LF, SF\_LA)$.

Right Shoulder Points

$$ {}^{f}P_1 = {}^{f}_{1}T\ R_{1x}R_{1y}R_{1z}\ {}^{1}_{4}T\ R_{4x}R_{4y}R_{4z}\,{}^{4}P_1 $$

$$ {}^{f}P_6 = {}^{f}_{1}T\ R_{1x}R_{1y}R_{1z}\ {}^{1}_{4}T\ R_{4x}R_{4y}R_{4z}\,{}^{4}P_6 $$

Where R_{4x} corresponds to a rotation of α_{4x} about system 4's X axis, R_{4y} corresponds to a rotation of α_{4y} about system 4's Y axis, and R_{4z} corresponds to a rotation of α_{4z} about system 4's Z axis. Additionally, ${}^4P_1 = [0\ 0.03 * SF\ -0.07\ 1]'$ and ${}^4P_6 = [0\ -0.03 * SF\ 0.04 + 0.03(SF\_H - 1)\ \ 1]'$, and $SF\ =\ max(SF\_T, SF\_RS)$.

Right Arm Points

$$^fP_2 = {}^f_1T\ R_{1x}R_{1y}R_{1z}\,{}^1_4T\ R_{4x}R_{4y}R_{4z}\,{}^4_5T\ R_{5x}\,{}^5P_2$$

$$^fP_5 = {}^f_1T\ R_{1x}R_{1y}R_{1z}\,{}^1_4T\ R_{4x}R_{4y}R_{4z}\,{}^4_5T\ R_{5x}\,{}^5P_5$$

$$^fP_3 = {}^f_1T\ R_{1x}R_{1y}R_{1z}\,{}^1_4T\ R_{4x}R_{4y}R_{4z}\,{}^4_5T\ R_{5x}\,{}^5P_3$$

$$^fP_4 = {}^f_1T\ R_{1x}R_{1y}R_{1z}\,{}^1_4T\ R_{4x}R_{4y}R_{4z}\,{}^4_5T\ R_{5x}\,{}^5P_4$$

R_{5x} corresponds to a rotation of α_{5x} about system 3's X axis. ${}^5P_2 = [0\ -0.03 * SF\ 0\ 1]'$, ${}^5P_5 = [0\ 0.03 * SF\ 0\ 1]'$, ${}^5P_3 = [0\ -0.03 * SF\_RF\ -0.02\ 1]'$, and ${}^5P_4 = [0\ 0.03 * SF\_RF\ -0.02\ 1]'$, where $SF\ =\ max(SF\_RF, SF\_RA)$.

Head and Neck Points

$$^fP_1 = {}^f_1T\ R_{1x}R_{1y}R_{1z}\ {}^1P_1$$

$$^fP_{12} = {}^f_1T\ R_{1x}R_{1y}R_{1z}\ {}^1P_{12}$$

$$^fP_2 = {}^f_1T\ R_{1x}R_{1y}R_{1z}\,{}^1_6T\ R_{6x}R_{6y}R_{6z}\,{}^6P_2$$

$$^fP_{11} = {}^f_1T\ R_{1x}R_{1y}R_{1z}\,{}^1_6T\ R_{6x}R_{6y}R_{6z}\,{}^6P_{11}$$

$$^fP_3 = {}^f_1T\ R_{1x}R_{1y}R_{1z}\,{}^1_6T\ R_{6x}R_{6y}R_{6z}\,{}^6P_3$$

$$^fP_{10} = {}^f_1T\ R_{1x}R_{1y}R_{1z}\,{}^1_6T\ R_{6x}R_{6y}R_{6z}\,{}^6P_{10}$$

$$^fP_4 = {}^f_1T\ R_{1x}R_{1y}R_{1z}\,{}^1_6T\ R_{6x}R_{6y}R_{6z}\,{}^6P_4$$

$$^fP_9 = {}^f_1T\ R_{1x}R_{1y}R_{1z}\,{}^1_6T\ R_{6x}R_{6y}R_{6z}\,{}^6P_9$$

$$^fP_5 = {}^f_1T\ R_{1x}R_{1y}R_{1z}\,{}^1_6T\ R_{6x}R_{6y}R_{6z}\,{}^6P_5$$

$$^fP_8 = {}^f_1T\ R_{1x}R_{1y}R_{1z}\,{}^1_6T\ R_{6x}R_{6y}R_{6z}\,{}^6P_8$$

$$^fP_6 = {}^f_1T\ R_{1x}R_{1y}R_{1z}\,{}^1_6T\ R_{6x}R_{6y}R_{6z}\,{}^6P_6$$

$$^{f}P_7 = {}^{f}_{1}T\; R_{1x}R_{1y}R_{1z}\; {}^{1}_{6}T\; R_{6x}R_{6y}R_{6z}\, {}^{6}P_7$$

Where R_{6x} corresponds to a rotation of α_{6x} about system 6's X axis, R_{6y} corresponds to a rotation of α_{6y} about system 6's Y axis, and R_{6z} corresponds to a rotation of α_{6z} about system 6's Z axis. Additionally, ${}^{6}P_1 = [0\ 0.1 * SF\_W\ 0\ 1]'$, ${}^{6}P_{12} = [0\ -0.1 * SF\_W\ 0\ 1]'$,
${}^{6}P_2 = [0\ 0.1 * SF\_T\ -0.035\ 1]'$, ${}^{6}P_{11} = [0\ -0.1 * SF\_T\ -0.035\ 1]'$,
${}^{6}P_3 = [0\ 0.035 * SF\_N\ 0\ 1]'$, ${}^{6}P_{10} = [0\ -0.035 * SF\_N\ 0\ 1]'$,
${}^{6}P_4 = [0\ 0.035 * SF\_H\ 0.08\ 1]'$, ${}^{6}P_9 = [0\ -0.035 * SF\_H\ 0.08\ 1]'$,
${}^{6}P_5 = [0\ 0.04 * SF\_H\ 0.08\ 1]'$, ${}^{6}P_8 = [0\ -0.04 * SF\_H\ 0.08\ 1]'$,
${}^{6}P_6 = [0\ 0.04 * SF\_H\ 0.28 + 0.07(SF\_H - 1)\ 1]'$,
${}^{6}P_7 = [0\ -0.04 * SF\_H\ 0.28 + 0.07(SF\_H - 1)\ 1]'$,

References

[1] Kruger, J., Lien, T. K., and Verl, A., 2009, "Cooperation of Human and Machines in Assembly Lines", *CIRP Journal of Manufacturing and Technology, 58(2), 628-646.*

[2] Nicora, M.L., Ambrosetti, R., Wiens, G.J., and Fassi, I., 2020, "Human-Robot Collaboration in Smart Manufacturing: Robot Reactive Behavior Intelligence", *ASME Manufacturing Science and Engineering Conference*, MSEC2020-10274, Cincinnati, OH, June 22-26.

[3] Streitmatter, G., and Wiens, G. (2020). Human-Robot Collaboration: A Predictive Collision Detection Approach for Operation within Dynamic Environments. *Manuscript submitted for publication.*

[4] Streitmatter, G., and Wiens, G. (2020). Human Modeling for Efficient Predictive Collision Detection . *Manuscript submitted for publication.*

[5] Yin, Z., Leu, M., Wiens, G., and Gao, R., 2018, "NRI: INT: COLLAB: Manufacturing USA: Intelligent Human-Robot Collaboration for Smart Factory", *2018 NSF/National Robotics Initiative (NRI) Principal Investigators' Meeting,* Arlington, VA, Oct. 29-30. poster (invited).

[6] Jiménez, P., Thomas, F., and Torras, C., 2001, "3D Collision Detection: A Survey", *Computers & Graphics,* 25(2), pp. 1-7.

[7] Culley, R. K., and Kempf, K. G., 1986, "A Collision Detection Algorithm based on Velocity and Distance bounds", *IEEE Int. Conf. on Robotics and Automation,* San Francisco, CA, 2, pp. 1064-1066.

[8] Abrams, S. and Allen, P. K., 2000, "Computing Swept Volumes", *Journal of Visualization and Computer Animation,* 11(2), pp. 69-82.

[9] Mamou, K. and Ghorbel, F., 2009, "A Simple and Efficient Approach for 3D Mesh Approximate Convex Decomposition", *IEEE International Conference on Image Processing,* ICIP V9, pages 3501–3504.

[10] Gaschler, A., Petrick, R. P. A., Kroger, T., Knoll, A., and Khatib, O., 2013, "Robot Task and Motion Planning With Sets of Convex Polyhedra", *RSS Workshop on Combined Robot Motion Planning and AI Planning for Practical Applications,* pp. 1-5.

[11] Erdim, H. Ilieş, and H. T., 2008 "Classifying Points for Sweeping Solids", *Computer-Aided Design,* 40(9), pp. 987–998.

[12] Ilies, H. T., 2009, "Continuous Collision and Interference Detection For 3D Geometric Models", *J. Comput. Inf. Sci. Eng.*, 9(2) pp. 1-7.

[13] Mainprice, J., and Berenson, D., 2013, "Human Robot Collaboration Manipulation Planning Using Early Prediction of Human Motion", *International Conference on Intelligent Robots and Systems,* Tokyo, Japan, Nov 3-7.

[14] Farin, G. and Hansford, D., 1999, "Discrete Coons Patches", *Computer Aided. Gerom. Design, 16, 691-700.*

[15] Crane, C. III, Duffy, J., 1998, *Kinematic Analysis of Robot Manipulators,* Cambridge University Press, New York, NY, Chapter 4, pp. 39-43.